dataCleaning

April 4, 2018

In [2]:
```
"""Global functions and variables"""

import pymysql

def open_conn():
    """open the connection before each test case"""
    conn = pymysql.connect(user='public', password='ece656yelp',
                                    host='maindb.czbva1am4d4u.us-east-2.rds.amazonaws.com
                                    database='yelp_db')
    return conn

def close_conn(conn):
    """close the connection after each test case"""
    conn.close()

def executeQuery(conn, query, commit=False):
    """ fetch result after query"""
    cursor = conn.cursor()
    query_num = query.count(";")
    if query_num > 1:
        for result in cursor.execute(query, params=None, multi=True):
            if result.with_rows:
                result = result.fetchall()
    else:
        cursor.execute(query)
        result = cursor.fetchall()
    # we commit the results only if we want the updates to Can't leave a review dated be
    # to persist.
    if commit:
        conn.commit()
    else:
        conn.rollback()
    # close the cursor used to execute the query
    cursor.close()
    return result

yelp_conn = open_conn()
```

0.1 Part I. Data cleaning

1. Check that no review is from the future or before Yelp's founding

```
In [13]: query_1 = "SELECT id, date FROM review WHERE unix_timestamp(date) <= unix_timestamp('20
                    OR unix_timestamp(date) >= unix_timestamp('2018-01-01')";

         result_1 = executeQuery(yelp_conn, query_1)
         result_1
```

```
Out[13]: (('03B9-gqbeGoMmPJbNzNT5w', datetime.datetime(2004, 9, 15, 0, 0)), ('PbIY2aIyszb6he6J-e
```

This shows 2 accounts that were created before Yelp's founding in October 2004.

2. Can't leave a review dated before account creation

```
In [2]: query_2 =  "SELECT user.id, user.yelping_since AS Date_started_yelping, review.date AS D
                    FROM (user INNER JOIN review ON user.id = review.user_id)\
                    WHERE user.yelping_since > review.date\
                    GROUP BY user.id;"

        result_2 = executeQuery(yelp_conn, query_2)
```

```
In [3]: len(result_2)
```

```
Out[3]: 191
```

This means these users somehow posted a review before their account was created, suggesting a glitch with their database.

```
In [5]: output = [print(result) for result in result_2[:20]]
```

```
('-58CWJ48is4duXgpvsWEGA', datetime.datetime(2013, 9, 18, 0, 0), datetime.datetime(2008, 10, 23,
('-9NfX8JO_5UVN_h1K8yOcg', datetime.datetime(2015, 2, 12, 0, 0), datetime.datetime(2010, 2, 26,
('-kEsfYKPs1_rgEWEIui2Mw', datetime.datetime(2015, 2, 14, 0, 0), datetime.datetime(2014, 4, 7, 0
('-KP8Me2KRqO7IwKIaFL-Vg', datetime.datetime(2013, 10, 19, 0, 0), datetime.datetime(2013, 9, 2,
('09T8OU8BDhQkiU8m4vZy_A', datetime.datetime(2013, 10, 21, 0, 0), datetime.datetime(2013, 10, 16
('0xjJDvZ6gZVoWRFEZJ48wA', datetime.datetime(2007, 1, 17, 0, 0), datetime.datetime(2007, 1, 15,
('1F9di6oPHhQm1qjZIcbsYA', datetime.datetime(2013, 12, 5, 0, 0), datetime.datetime(2013, 10, 5,
('2Ea6wAkeOPyZ7BD-0rPejQ', datetime.datetime(2013, 9, 18, 0, 0), datetime.datetime(2012, 10, 9,
('2oxUNDpouxH8Y02yG6pG-w', datetime.datetime(2010, 4, 28, 0, 0), datetime.datetime(2006, 1, 25,
('37jJedy6_ptCmNvBJ-H54g', datetime.datetime(2014, 8, 16, 0, 0), datetime.datetime(2007, 3, 10,
('3BHeHTGNZcaEi8woqGy2kA', datetime.datetime(2013, 10, 19, 0, 0), datetime.datetime(2013, 10, 4,
('3cKcTbCIWp97pK8lo0Ao5Q', datetime.datetime(2015, 2, 12, 0, 0), datetime.datetime(2012, 11, 4,
('3M1IB2QtJOuBlneiZ1532w', datetime.datetime(2013, 10, 22, 0, 0), datetime.datetime(2013, 10, 4,
('3wCR1kGccoxgfuzzQ0_p-A', datetime.datetime(2013, 10, 18, 0, 0), datetime.datetime(2012, 1, 21,
('4jIlAyittJukBRteYp2Gig', datetime.datetime(2014, 1, 28, 0, 0), datetime.datetime(2011, 12, 20,
('5eLIdrtZFjQO1VCWwWok2w', datetime.datetime(2013, 8, 18, 0, 0), datetime.datetime(2012, 8, 13,
('5o3iqx2ueZmiTIRVuuO4nw', datetime.datetime(2013, 5, 26, 0, 0), datetime.datetime(2012, 12, 24,
('6ESU4WqmScR1Vpjr1nDDig', datetime.datetime(2013, 10, 17, 0, 0), datetime.datetime(2013, 10, 14
('6i8vhqI1x9qzJ4fjT_x6xQ', datetime.datetime(2008, 3, 24, 0, 0), datetime.datetime(2007, 3, 19,
('6LDypE52IBrhXeYsd-6uqw', datetime.datetime(2015, 2, 14, 0, 0), datetime.datetime(2014, 1, 10,
```

3. Can't be elite in a year before their account was made

```
In [26]: query_3 =  "SELECT user.id, user.yelping_since AS Date_of_yelping, elite_years.year AS
                     FROM (user INNER JOIN elite_years ON user.id = elite_years.user_id)\
                     WHERE YEAR(user.yelping_since) < elite_years.year\
                     GROUP BY user.id;"

         result_3 = executeQuery(yelp_conn, query_3)

In [27]: len(result_3)

Out[27]: 55493
```

4. Can't checkin outside open hours

```
In [33]: query_4 = "SELECT COUNT(*) FROM checkin JOIN (SELECT hours.business_id, SUBSTRING_INDEX
                    AS day_of_week, SUBSTRING_INDEX(SUBSTRING_INDEX(hours, '|', - 1), '-', 1) AS
                    SUBSTRING_INDEX(SUBSTRING_INDEX(hours, '|', - 1), '-', - 1) AS closing_time
                    AS a ON a.business_id = checkin.business_id\
                    AND a.day_of_week = SUBSTRING_INDEX(checkin.date, '-', 1)\
                    WHERE a.opening_time > SUBSTRING_INDEX(checkin.date, '-', - 1)\
                    AND a.closing_time < SUBSTRING_INDEX(checkin.date, '-', - 1);"
         result_4 = executeQuery(yelp_conn, query_4)
         result_4[0]

Out[33]: ((480488,),)
```

This shows that there are many check ins that occur outside of the businesses open hours which are potentially invalid checkins, but this is not a guarantee because it is possible that the business changed their hours of operation at some point after someone checked in, resulting in the discrepency.

This also shows the need within the database for the date column in the checkin and the hours column in the hours table to be normalized by splitting into date, opening time and closing time columns as this would save computation time having to perform substring_index computations on every row.

5. `User.review_count` **cannot be less than the sum of the number of reviews by a user**

```
In [17]: query_5 = "select count(*) from user join (select count(user_id) as countedReviews, use
                    review group by user_id) as a on a.user_id = user.id where a.countedReviews
         result_5 = executeQuery(yelp_conn, query_5)
         print(result_5[0])

(1319,)
```

This shows that the way Yelp gets the review_count number is potentially flawed since it should never count there being less reviews than the number of reviews provided for each user, unless the dataset that it was acquiring the count from was out of date.

6. Cannot be Elite in an invalid year Invalid years include ones before 2004, years in the future or years they didn't post a review, tip or photo.

```
In [12]: query_6 = "select count(*) from elite_years join \
         (SELECT user_id, SUBSTRING_INDEX(date, '-', 1) AS year FROM review) \
         as a on a.user_id=elite_years.user_id and a.year = elite_years.year \
         group by elite_years.user_id, elite_years.year;"
         result_6 = executeQuery(yelp_conn, query_6)
         print(result_6[0])
```

```
(37,)
```

In total there are 186900 entries in elite_years. 37 appear to be erroneous.

0.2 Part 2. Data Indexing

To allow for faster queries additional indices were added to some tables in the database. This includes: * adding one to the year column in the table elite_years to speed up query_6 * create index idx_year on elite_years(year) * adding one to the yelping_since column in the user table * create index idx_yelping_since on user(yelping_since) * creating new columns in the hours table to split the hours column into date, opening_time and closing_time and creating an index for each of these * create index idx_day_of_week on hours(day_of_week) * creating new columns in the checkin table to split the date column into date_of_week and time and creating an index for each of these * create index idx_day_of_week on hours(day_of_week) * creating new columns in the review table to split the date column into year, month, day columns and scrapping the time because its always 00:00:00 * select count(*) from review where substring_index(date, ' ', -1) != '00:00:00'; #returns 0